**PATENT APPLICATION**
**VIXS.0100010**

**IN THE UNITED STATES PATENT AND TRADEMARK OFFICE**

**FILING OF A UNITED STATES PATENT APPLICATION**

# DEVICE AND METHOD FOR COMPRESSION OF A VIDEO STREAM

**INVENTORS:**

**Indra Laksono**
**138, Old Hill Street**
**Richmond Hill, Ontario**
**Canada**
**L4C 9Z7**

**ATTORNEY OF RECORD**
**J. GUSTAV LARSON**

**SIMON, GALASSO & FRANTZ, PLC**
**P.O. Box 26503**
**Austin, TX 78755-0503**
**PHONE (512) 336-8957**
**FAX (512) 336-9155**

Express Mail Label No. **EL855710810US**

Date of Deposit: 3-27-01

I hereby certify that this paper is being deposited with the U.S. Postal Service "Express Mail Post Office to Addresses" service under 37 C.F.R. Section 1.10 on the 'Date of Deposit', indicated above, and is addressed to the Commissioner of Patents and Trademarks, Washington, D.C. 20231.

Name of Depositor: **Terri Alloway**

(print or type)

Signature: Terri Alloway

# DEVICE AND METHOD FOR COMPRESSION OF A VIDEO STREAM

## FIELD OF THE DISCLOSURE

This invention relates generally to processing video streams, and more particularly to recompression of downscaled decompressed video streams.

## BACKGROUND

Continuous video stream playback systems using MPEG and variants are well known in the art. With improvements in memory subsystems and digital content delivery methods, higher continuous video images, such as those using the HDTV/ATSC 1920x1080i format will become more common.

A continuous moving video image stream in its raw form requires very high transmission rates. One way of limiting by the high transmission rates is currently solved by using compression schemes, such as MPEG encoding schemes, that take advantage of continuity in inter-frame content to create very highly packed data. MPEG and similar variants use motion estimation of blocks of image data between frames to perform this compression.

The step to compress video data is processor and memory bandwidth intensive. Motion estimation, a compression step, requires a large amount of the computational work that can use up a significant amount of available bandwidth. In Motion Estimation, a frame of image data is first subdivided into a plurality of fragments or blocks. Next, a fragment or group of blocks of the current frame image is compared against one or more fragments or group of blocks in another frame or frames. The optimal fragment or group of blocks in each of the alternate frames may not be in the same location as the current frame. This location is often different between each of the alternate frame or frames and the current frame. The location of each of these fragments is represented as a motion vector of the form (+/- Dx, +/-Dy).

In the case of MPEG, each of Dx and Dy is a multiple of 0.5. A motion vector that has a 0.5 component means that the fragment from an alternate frame is formed by averaging two side by side pixels from either the same horizontal row or vertical column or four pixels in a 2x2 block. A complex processor and memory bandwidth intensive search algorithm that has to consider a plurality of fragments combinations is generally used to construct each motion vector.

With extremely high resolutions, such as the 1920x1080i format, the data rate of such a compressed stream will be very high. This high data rate poses at least three sets of problems. First, to record or save such a stream over any length of time requires large amounts of storage that can be prohibitively expensive. Second, many display devices that can be used to view such a stream may not be capable of displaying such a high resolution data stream. Third, where there is a data network with multiple viewing or receiving devices, such a network will typically have a fixed bandwidth or capacity. Such a network may be physically incapable of simultaneously supporting multiple viewing devices.

Note, that in general, there can be a plurality of motion vectors required to build each fragment or macroblock of a frame. This further adds to the processing and bandwidth problem. Accordingly, there is a need for an improved device and method for processing video streams.

## BRIEF DESCRIPTION OF THE DRAWINGS

Fig. 1 is a functional block schematic diagram of an MPEG transcoder processor in accordance with one embodiment of the present invention.

Fig. 2 is a detailed flowchart illustrating operation of the MPEG transcoder processor of Fig. 1.

## DETAILED DESCRIPTION OF THE FIGURES

Referring to Fig. 1, a high level functional block diagram of a video processing device is shown. The video processing device has an input buffer 100 that receives data representing a video image to be provided to a general MPEG decoder 110. The MPEG decoder 110 creates either the video image or an optionally scaled down representation of the video image decompressed frames in a memory, RAM1 120, and also obtains the motion vectors from the decompression process. The motion vectors are saved in memory, RAM2 140. Note, the motion vectors are not normally saved in memory using a standard MPEG decoder. Also note that in other embodiments, the memories 120 and 140 can include dynamic random access memory, static dynamic random access memories, hard drive storage devices, and the like.

The MPEG encoder 150 performs many of the steps of normal MPEG encoding, but avoids the computationally intensive motion estimation step by retrieving the motion vectors previously saved in memory, RAM2 140. By retrieving the set of surrounding motion vectors and building a new set of motion vectors, the encoder 150 avoids the expensive search that is required in conventional motion estimation. The output buffer 160 is a bit bucket that accumulates data bits before it out to a final destination, such as memory or any output ports to a device coupled to receive such data.

With reference to the specific embodiment of Fig. 2, elements with labels from 310-395 indicate functions the decoder portion, and elements 410-495 identify functions encoder portion of the transcoder. Note, that this example assumes an image downscale of ½ x ½. A macroblock, in MPEG terminology, is a 16x16 matrix of individual picture elements. A block in MPEG terminology is an 8x8 matrix of individual picture elements. When downscaling by ½ x ½, it is assumed that a 2x2 set macroblocks are converted to form a single macroblock. The ½ x ½ downscaling operation is typically performed with an effort to preserve as much content of the original image as possible, while presenting the final result in a smaller bitmap. Downscaling is well understood to mean a process where a group of picture elements are combined in some fashion to create another group consisting of less picture elements. For downscaling of ½ x ½ several options are available. For example, one possible implementation, the picture elements are blended in a predefined method. However, one reasonably versed in the art will understand that there are multiple ways to blend them to achieve the same results or perhaps to even scale without blending.

Referring again to Fig. 2, the bit parsing step 310 retrieves a current macroblock being decoded. Variable length decode 320, for example using DeHuffman, is performed, and can be run level or run length decoding, to retrieve information about a particular block. The information retrieved contains a series of run-level sequences, each representing an 8x8 spare matrix (known as a block) with a run of sequential zero values implicitly defined by the run. When referred to as run-level within the MPEG scope, the run refers not to repeated runs of the value in level, but to consecutive runs of 0. In the case of MPEG, the block is built in a zigzag fashion.

After the decode, the process dequantizes the generated block at 330, which involves multiplying each element in the block with an element in a matrix. As is well known in the art, dequantization is inverse quantization, where a matrix of fixed constants might be retrieved or determined at the start of the decoding process, rarely changing as the stream is decoded. Each element of the sparse matrix is multiplied with this constant value from the corresponding entry in this matrix of constants. To understand the significance of the inverse quantization process, it should be understood first that during the compression process, the blocks went through a discrete cosine transform (DCT) step to convert the picture blocks into the frequency domain. In this representation of the picture blocks, the original blocks can still be retrieved (up to arithmetic rounding resulting from limitations in number representation) without any loss.

In the frequency domain, the block of data has an interesting property. The main determinants of the appearance of the image to the human eye is primarily decided by the terms of the block (matrix) that are in the top left corner (starting at indices [0,0,] of the matrix). Changes to the terms to the bottom right of the block tend to have less of a visible effect on the reconstructed blocks to the human eye. The purpose of quantization during the encoding process is to take advantage of this property and attempt to treat terms that are close to zero and positioned closer to the bottom right, as zero, while preserving as much of the information in the top left corner as possible.

After the block has been dequantized, we proceed to the inverse Discrete Cosine Transform, (iDCT), step 340, that obtains the block in its raw form. The steps 310-340 proceed via a loop through the end of macroblock decision step 350, until a complete macroblock is obtained. For MPEG, this macroblock typically consists of 4 (2x2) blocks of information in the Y (luma, or brightness) and 1 block of Cr and 1 block of Cb. When the complete macroblock is obtained, the process proceeds to test decoder motion vector step, at 360. When there is no motion vector for a particular macroblock, the macroblock will be downscaled for example by either 1,2,4 or 8 and written out during the downscale and write macroblock step, at 390.

Where there is a set of motion vectors, the motion vectors will be saved, at step 370, into a storage area, such as memory 140 of FIG. 1, that holds all the original motion vectors used to build this frame. Motion compensation, at step 380, is performed to build a new macroblock. This new macroblock is then downscaled by either 1,2,4 or 8 and provided to in the downscale and write macroblock step, at 390.

Sub B1

~~When the process reaches the decoder end of frame step, at 395, if the frame has finished, the process initializes the frame encoder, at 410, which will start to encode a macroblock, at 420. If the current macroblock has no motion vectors (determined at step 430), then the macroblock is read, at step 470, from the downscaled and decompressed frame created during the decoding process, and each block in the macroblock undergoes a discrete cosine transform, at step 480. If the current macroblock has motion vectors (determined at decision step 430), the four sets of neighboring motion vectors are retrieved from storage, at 440, and are used to build the original image frame, at steps 450 and 460. In this example, note that scaling of ½ x ½, is used. Retrieval of more motion vectors for other scale factors would be required. For example, if scaling by 1/3 x 1/3, 9 motion vectors would be used. If scaling is by 2/5 x 2/5, between 4 to 9 motion vectors would be used, depending on how the resultant motion vector is generated.~~

Sub B2

~~The new motion vector, at 450, can be built in multiple ways. In one method, one may choose to use a simple averaging modulo ½ of each component of the vectors from each of the four sets of motion vectors. In an alternate embodiment, one may choose the most frequently occurring motion vector ($\Delta X_k$, $\Delta Y_k$) from each set of kth-motion vectors, with an arbitrary method for~~

breaking ties. One possible ~~method of breaking ties is to~~ choose the element that is closest to the top left ~~motion~~ vector.

With the new motion vector built at step 450, the process proceeds to read the macroblock from the stored decompressed image frame and then builds a delta frame containing the result of applying a reverse motion compensation step, to obtain the delta macroblock, at 460. At this point, the delta macroblock is sent to the unit that performs DCT on all the blocks of the macroblock, at 480. The resulting transformed block undergoes quantization, at 485 (rounding integer division of each element by elements of a matrix). The resulting quantized matrix representation of each block is then variable length encoded, at 488, and the compressed result is sent to the output encoded macroblock unit, at 490. This process continues until detecting the encoder end of frame, at 495, signaling the decoder to begin working on the next frame.

Note, with respect to Fig. 2, that with use of double buffering of the motion vectors, and queuing other command streams, both the encoder and decoder steps may both run in parallel.

One feature of the disclosed embodiment is that, where there are motion vectors, the motion vectors will be saved, at 370, into a storage area that holds all the original motion vectors used to build the processed frame. In addition, the compute intensive and costly motion estimation step is avoided by retrieving the stored motion vectors and building a new motion vector set using simple process steps, at 450. By avoiding the expensive motion estimation step, the disclosed embodiment may provide a much more cost-effective solution that achieves largely the same quality of transcoding as systems that searches the motion vectors from scratch.

The particular embodiments disclosed herein are susceptible to various modifications and alternative forms. Specific embodiments therefore have been shown by way of example in the drawings and detailed description. It should be understood, however, that the drawings and detailed description are not intended to limit the invention to the particular form disclosed, but on the contrary, to the maximum extent permitted by law, the invention is to cover all modifications, equivalents and alternatives falling within the spirit and scope of the present invention as defined by the following claims, and their equivalents. For example, in the system illustrated in FIG. 2, the connections between the decoder 110 and the memories 120 and 140 may represent separate busses or common busses. Likewise, the connection between the encoder 150 and the output buffer and the connection between the decoder and the input buffer may represent the same or different connections, and may even be common with the connections to the memories 120 and 140. Also, in another embodiment of the present invention, one of a normal mode of operation, where the encoder determines the motion vectors, and a motion vector reuse mode as described herein is selected. Selection of one mode will generally be based on the availability of previously saved motion vectors. Note that in another embodiment, during a normal mode of operation, the decoder would not save motion vectors.